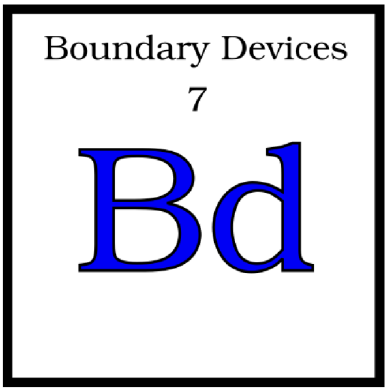
User's Manual for the Boundary Devices Nitrogen® board

July 6, 2010



1 Revision History

Date	Revision	Description
2010-06-17	1.0	First (rough) draft
2010-06-25	1.1	Still rough. Additional details on Ubuntu Live images
2010-07-01	1.2	Added connector diagram
2010-07-06	1.2	Refined connector diagram, added U-Boot notes, removed Ubuntu build notes

Contents

1 Revision History **2**

2 Intended Audience **5**

3 Overview of features **5**

4 Hardware feature **6**

4.1 Layout 6

4.2 Mounting 7

5 Software features **8**

5.1 Internal ROM boot loader and boot flow 8

5.2 Das U-Boot 9

5.2.1 Using U-Boot with Linux 9

5.2.2 Using U-Boot with Windows CE 11

5.3 Windows CE 12

5.4 Linux Kernel 13

5.4.1 Kernel configurations 13

5.4.2 Kernel compilation for the impatient 13

5.5 Linux Toolchains 13

5.6 Busybox 13

5.7 Ubuntu 14

5.8 Linux display setup 15

2 Intended Audience

This document aims to provide the information needed to integrate the Nitrogen® board into your application. As such, it addresses both hardware and software integration.

Note that this manual contains many references to outside projects which have a life of their own, so it should generally be used as a starting point for how a Nitrogen® may be used.

Please contact Boundary Devices with any questions.

3 Overview of features

The following are highlights of the Nitrogen® board.

- Freescale i.MX515 800MHz ARM Cortex A8
- Freescale MC13892VL Power Management and User Interface IC
- Up to 512MB DDR2 Memory
- Full featured Boot Loader for custom startup
- Board Dimensions: 4" x 2.5"
- Up to 8MB Serial Flash
- Directly interfaces to 4.3" and 7" LCD Displays
- LED Backlight Driver Circuitry
- 44 KHz Stereo 16-bit Audio Output
- 44 KHz Monaural Audio Input (microphone)
- 3 RS-232 Serial ports
- 1 High-Speed USB 2.0 Master Port
- 1 High-Speed USB 2.0 OTG
- Built-in 5MP Camera Support
- Touch-Screen Support
- SPI/I2C Interface
- microSD Slot for Expanded Storage
- General Purpose I/O for Device Control
- Dry Contact Output
- On-board interface to Symbol 1D/2D Barcode Scanner
- Supports Windows CE, Ubuntu Linux, or Android Operating Systems
- Customized Versions Available

4 Hardware feature

4.1 Layout

As shown in Figure 1, the Nitrogen® board contains a wide variety of I/O options for use in your application. Note that some of these may not be populated on an evaluation or production board.

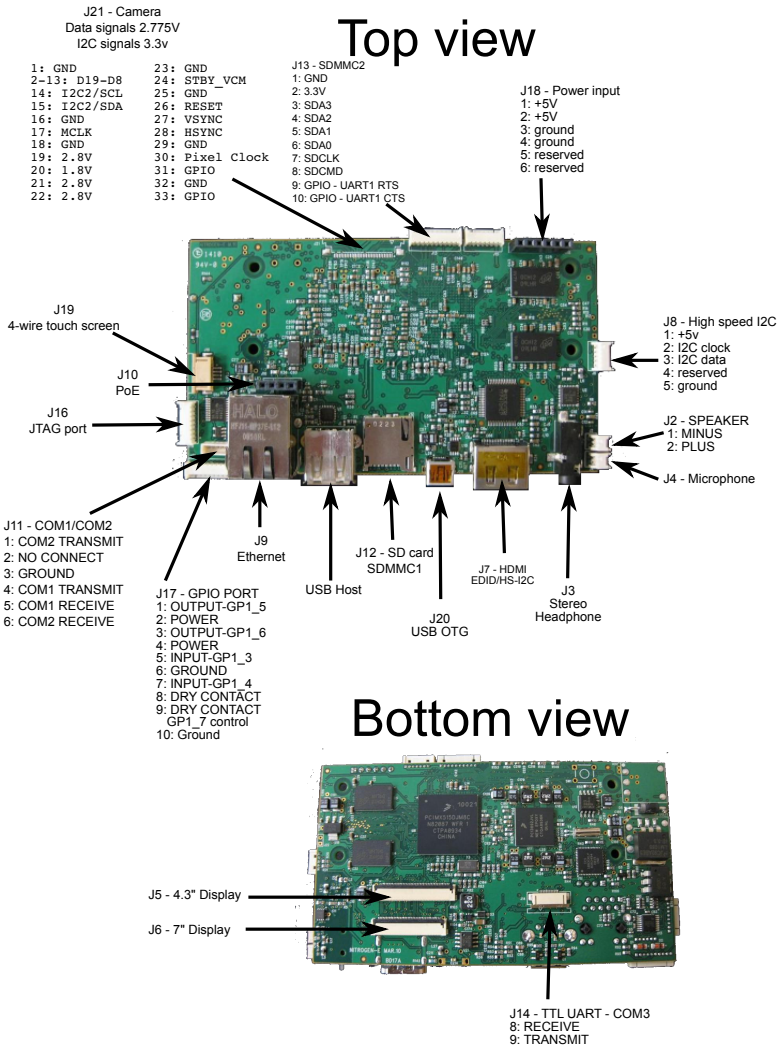


Figure 1: Nitrogen board

4.2 Mounting

Including protrusions for connectors, the Nitrogen® board measures 4.5" by 2.5", the same size as the Hydrogen board. This allows the use of both the 7" and 4.3" enclosures with VESA-mounts as well as the Kiosk and in-wall enclosures.

The board is designed to be mounted using the four mounting holes as shown in figure 2.

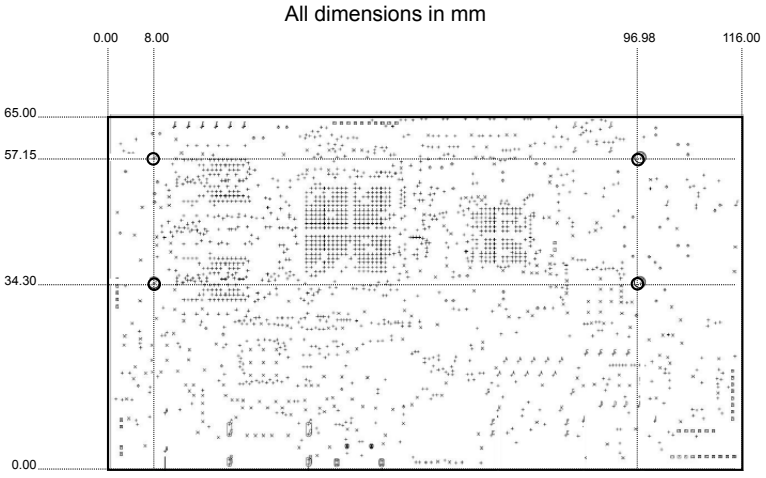


Figure 2: Nitrogen-E mounting

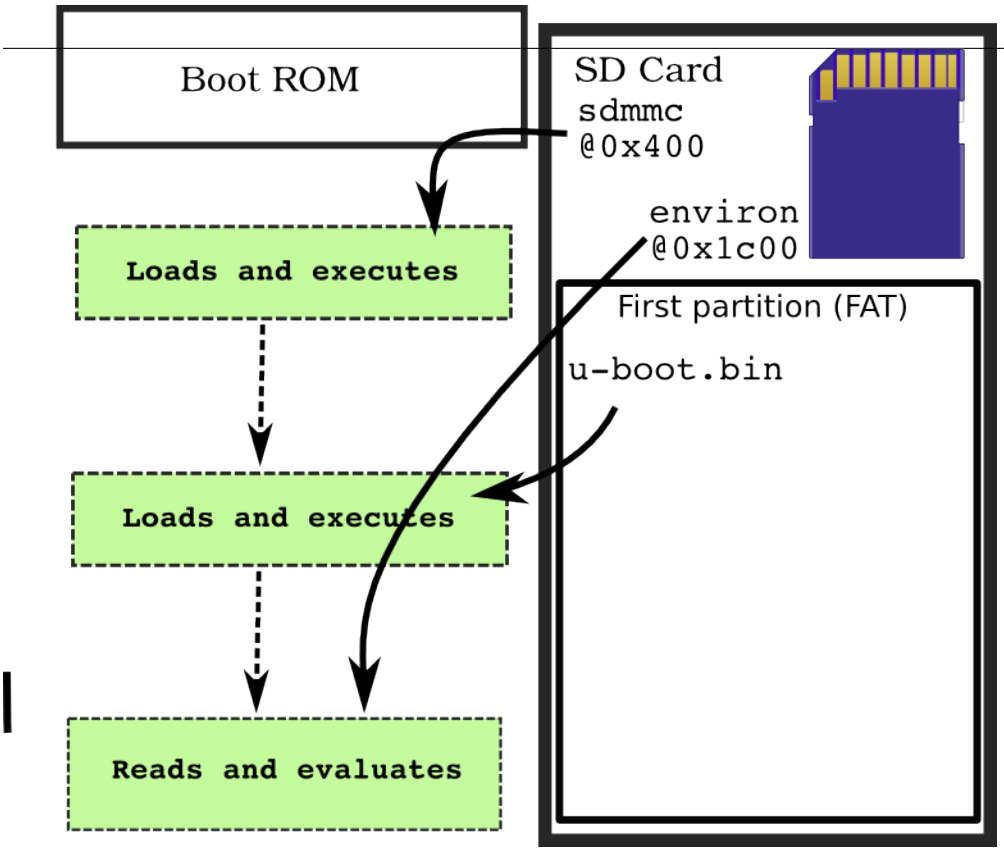


Figure 3: Nitrogen boot flow

5 Software features

As provided by Boundary Devices, the Nitrogen® board supports either the Windows CE 6® or Linux operating systems. There are currently two shipping versions of Linux:

1. **Busybox** - A small Linux userland, and
2. **Ubuntu** - A full-featured X-Windows based installation

To simplify the installation of either, the U-Boot boot loader is installed on our evaluation boards, and one or more MMC cards are shipped to allow the use of either operating system.

5.1 Internal ROM boot loader and boot flow

The i.MX51 processor contains an internal boot loader in ROM that supports boot from serial EEPROM, SD card, USB, NAND or NOR flash. Freescale has a number of documents about how this process is done. In this document, we'll describe how the demonstration images are currently configured.

To begin with, the Nitrogen® is configured to boot from SD card. As shown in figure 3, the internal ROM loads code from offset 0x400 (1k) and executes it. Note that this first set of code is placed in non file-system area, so it must be loaded to disk using a tool such as `dd` under Linux.

The current incarnation of U-Boot then loads its' environment settings from offset 0x1c00 (block 14) of the SD card. This is also in space before the first partition.

The next steps depend primarily on the values found in the environment, especially the `bootcmd` variable.

Note that future iterations of the Nitrogen® will likely place both U-Boot and the environment variable in serial EEPROM.

5.2 Das U-Boot

The U-Boot Boot Loader is a full-featured loader for either Linux or Windows CE that supports a wide variety of options for loading your Operating System and application.

The U-Boot Boot Loader is a very capable loader with support for USB and network boot, including BOOTP/DHCP, and NFS mounting support.

Please refer to the U-Boot website for details of the operation. The sections which follow will describe typical usage for various operating systems and variants.

In general, though, our O/S releases will have the following features.

Serial console Releases from Boundary Devices will be configured to allow access to the U-Boot command-line on the primary serial port[1] with a baud rate of `115200`, no parity, 8 data bits. They'll be configured to stop the boot process on any input character on the console.

`bootcmd` in persistent environment

The U-Boot variable `bootcmd` will be stored in persistent memory[2] and loaded at boot time. The content of this command will be executed if no character is received on the console within a time frame defined by the `bootdelay` variable.

Environment variables can be saved using the `saveenv` command.

```
U-Boot> set bootcmd 'mmcinit && fatload mmc 0 92000000 NK6.nb0 && go 92000000'
U-Boot> savee
U-Boot> print
```

In the U-Boot shell, single-quotes do not perform variable expansion of the quoted item, but double-quotes do.

The `print` command in U-Boot is used to display the content of all environment variables.

5.2.1 Using U-Boot with Linux

When using U-Boot to load Linux, there are two typical use cases for the `bootcmd` settings to boot with or without a RAM-disk. In general, we recommend the use of a RAM-disk whenever booting to SD card so that the RAM disk, or `initrd` can perform filesystem checking. The Freescale Ubuntu image is not currently set up to do so. If you're using an NFS root filesystem, you also won't want a RAM disk.

Generally, we'll set up `bootcmd` as follows when using a RAM disk:

```
U-Boot> set bootcmd 'mmcinit &&
```

[1] `COM1:` under CE, `/dev/ttymxc0` under Linux
[2] SD card or serial EEPROM

```
        fatload mmc 0 92000000 uImage &&
        fatload mmc 0 92400000 &&
        bootm 92000000 92400000'
U-Boot> saveenv
```

In English, this says:

- `mmcinit` - Initialize the SD card
- `&&` - If that works
- `fatload mmc 0 92000000 uImage` - Load uImage from SD card 0 fat filesystem to address 92000000
- `&&` - If that works
- `fatload mmc 0 92400000 initrd.u-boot` - Load RAM-disk (`initrd.u-boot`) at address 92400000
- `&&` - If that works
- `bootm 92000000 92400000` - Boot Linux with kernel at 92000000 and initrd at 92400000

To boot without a RAM disk, we'll just skip loading it and only supply a single argument to the `bootm` command. In this case, the filesystem root should be specified on the kernel command-line (`bootargs` U-Boot variable.

```
U-Boot> set bootcmd 'mmcinit &&
        fatload mmc 0 92000000 uImage &&
        bootm 92000000'
U-Boot> saveenv
```

Whenever you're booting Linux, these are some key kernel command-line variables you may want to set:

- `video=` - Specifies the video resolution and output form. See section 5.8 for details.
- `console=ttymxc0,115200` - Sets `/dev/console` to the first UART, the same as U-Boot uses

If you're booting over NFS, you'll need to add these clauses:

`ip=dhcp`	This tells the kernel to perform a DHCP to get an IP address. You'll also need kernel support for DHCP to use this. Check with the Linux command `zcat /proc/config.gz | grep DHCP`.
`rootwait`	This clause tells the kernel not to expect that a RAM disk is immediately available.
`root=/dev/nfs`	This clause tells the kernel that the root device is NFS.
`nfsroot=10.0.0.1:/path/to/rootfs`	This clause tells the NFS device driver what server and path to use as the root filesystem.

5.2.2 Using U-Boot with Windows CE

Using U-Boot to load Windows CE is simpler, since it uses neither `bootargs` nor a RAM disk and currently has compiled-in display settings.

Consequently, the command line is typically this to load CE from SD card:

```
U-Boot> setenv bootcmd 'mmcinit &&
        fatload mmc 0 90200000 NK6-nitrogen_e.nb0
        && go 90200000'
U-Boot> saveenv
```

If you want to load CE over TFTP, a suitable command-line might be this:

```
U-Boot> set bootcmd 'dhcp 90200000 192.168.0.251:NK6-nitrogen_e.nb0
        && go 90200000'
U-Boot> saveenv
```

Note that each of these is using the `go` command and not `bootm`.

5.3 Windows CE

As mentioned earlier, the Nitrogen® board ships with a runnable Windows CE 6.0 image on MMC card. A Board Support Package is also available and necessary to tailor the operating system for a given application.

5.4 Linux Kernel

The sources for the Linux kernel for Boundary Devices boards are available on our `git` server.

We're currently using branch `buntu` for Ubuntu support and branch `watchie6` for Busybox development.

We also supply the source code used to build a given kernel on SD cards in directory /linux-bd.

5.4.1 Kernel configurations

We currently use configuration `nitrogen_defconfig` for compiling Busybox kernels, and configuration `nitrogen_ubuntu_defconfig` for Ubuntu builds.

5.4.2 Kernel compilation for the impatient

For Busybox:

```
~/linux-bd $ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- nitrogen_defconfig

~/linux-bd $ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- uImage modules
```

For Ubuntu:

```
~/linux-bd $ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- nitrogen_ubuntu_defcon

~/linux-bd $ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- uImage modules
```

In each case, the U-Boot-wrapped kernel (`uImage`) is located in `arch/arm/boot`.

You can install the modules into an `initrd` or `nfs` filesystem by using the `modules_install` target and the `INSTALL_MOD_PATH` environment variable:

```
~/linux-bd $ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- \
             INSTALL_MOD_PATH=~/ubuntu-initrd/ \
             modules_install
```

5.5 Linux Toolchains

We're currently using CodeSourcery® 's toolchain `arm-2010q1` for compilation of our kernels and Busybox applications and the native compiler from Ubuntu.

5.6 Busybox

Describe Busybox platform here.

5.7 Ubuntu

Our Ubuntu builds are "Live" images so they're set up for read-only access to the SD cards. We put together a document, available on our web-site.

5.8 Linux display setup

Our current version of U-Boot does not have display support, but it can be used to configure the kernel's display until we get proper display support.

The `lcdpanel` command is currently used to define an environment variable `panel` that can be used to construct the kernel command line arguments[3]. The command contains support for most panels shipped by Boundary Devices as well as support for `Discrete Monitor Timings` (dmt) and the VESA Generalized Timing Formula.

It may be used in one of the following ways:

command string	description
lcdp ?	Show the list of currently supported lcd panels
lcdp panelname	Select and initialize a known panel by name
lcdp vesa:WxH@FREQ	Select and initialize a VESA GTF panel with specified Width, Height, and FREQu
lcdp +	Add a new panel (prompts for all of the details)
lcdp name:field,field...	Specify a panel name in all it's gory details

As mentioned earlier, this command does not currently configure the display adapter on the Nitrogen board. It simply sets the `panel` environment variable for use in constructing a `bootargs` variable.

To rehash previous comments, the `bootargs` variable contains the Linux kernel command-line.

The Linux kernel display driver contains support for initializing the display adapter through the use of the '`video=mxcfb:`' kernel parameter. To enable arbitrary displays, we added support for a `raw` specifier that matches our previous use of the `lcdpanel` U-Boot command.

To complicate matters, we also added support for three I/O widths to support the physical connections to two internal connectors and the HDMI adapter:

```
777 - 7" display connector (21-bit color)
666 - 4.3" display connector (18-bit color)
888 - HDMI output (24-bit color)
```

These I/O identifiers are added after the `panel` variable in a kernel command-line, separated by a comma.

Putting all of this together, the following example shows how to set the kernel command line for an Ubuntu Live boot with a 1024x768 panel over HDMI:

```
U-Boot> lcdp vesa:1024x768@60
U-Boot> set bootargs "video=$panel,888 boot=casper"
U-Boot> saveenv
```

To use a 7" panel in a Boundary enclosure, you can specify things like this:

```
U-Boot> lcdp vesa:800x480@60
U-Boot> set bootargs "video=$panel,777 boot=casper"
U-Boot> saveenv
```

[3] variable `bootargs`

To use the 4.3" panel, you can specify this (using a known panel type):

```
U-Boot> lcdp urt_480x272
U-Boot> set bootargs "video=$panel,666 boot=casper"
U-Boot> saveenv
```